Text Classification with pre-trained word embeddings

Javin Liu , Yuxuan Tian, Zhixin Xiong

McGill University

Abstract

In natural language processing (NLP), text classification is the problem of assigning categories for text data according to its content. And word embedding is a term used for the representation of words for text analysis. Pre-trained word embedding, which are trained on large datasets and then saved, can be utilized in deep learning models for solving other classification tasks. By using such a word embedding, it may help give a boost in performance to our model because they provide the model with more information to learn from over using regular one-hot encoding. We claim that different pre-trained word embedding's may have different performance on some highly specialized datasets since they are pre-trained on large datasets. Therefore, we conducted specific tasks with famous word2vec and cutting edge models such as BERT and experiment with a lstm-based classifier on different dataset to explore the affect of different embedding on different datasets. We also consider the impact of filtering words of low frequency and removing stop-words.

1 Introduction

Text analysis is an integral part in many fields. The automated classification of content has reduced a great amount of manual work and brought about huge economic benefits. For example, by analysing the users' comments on the review section of a product, the company is able to get an understanding of the general user sentiment over the product without manually searching over all the reviews.

A model that is commonly used for text classification is the LSTM model (Graves et al., 2005) as shown in figure 1. It is one of the most import models in NLP because it's unique "memory cell" (hidden layer vector) allows it to remember past info about the previous cells. Having such a memory allows it to be used on tasks related sequential data which in our case is sentences.

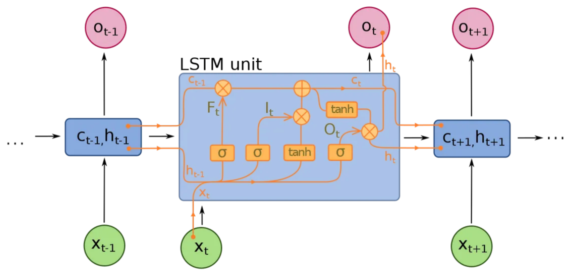

Figure 1: LSTM Model Structure

Because machines are unable to understand human language, we have to convert our training data into a more understandable form for the machine to train on. One of the most common methods is one-hot-encoding. It allows us to map every categorical word to an integer value. Similarly, embedding is also a technique that allows us to represent words and sentences in a vector form. This is because machines generally perfer to learn on numbers rather than characters. By using embedding we are able to represent word and sentences using unique vector representations. Embedding is a large part of NLP research and many methods have been imposed for this task. Such models include glove embedding(Pennington et al., 2014), Word2Vec Embedding (Goldberg and Levy, 2014), and the most recent BERT Model Embedding (Devlin et al., 2018). As explained on the following website, Word2vec using one neural network hidden layer to predict a target word rom its neighbors for a skip gram model or a word from its context for a CBOW. One of the limitation of word2vec is that we cannot have an embedding for words that are not in the vocabulary of the training data.

BERT is a model that is inspired by transformers. (Wolf et al., 2020). It is a bidirectional transformer-based language model that uses encoders and a revolutionary technique called masked language

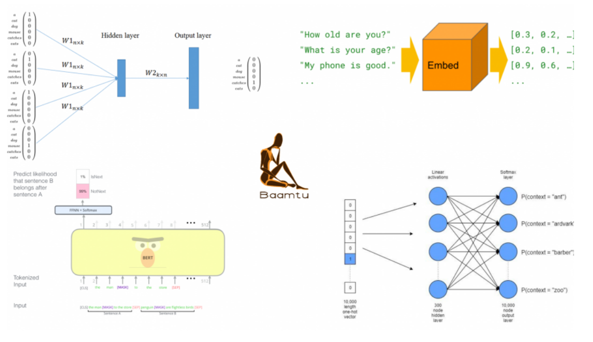

Figure 2: Embedding Illustrated

modeling. The BERT was able to outperform similar models and is an very common model being used today.

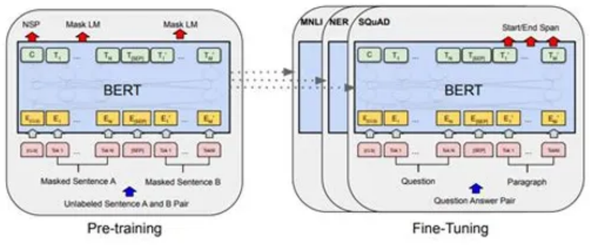

Figure 3: BERT Model Structure

Due to the profound amount of options for embedding and the vast amount of differences of these models, we would like to examine the different embedding techniques on several different data sets to examine the effects of these models on different data and see if certain design decision of these embedding techniques allow them to perform differently. We used our model to perform spam detection, sentiment analysis, mood detection, and sentiment prediction of tweets about covid.

2 Related Work

Embedding is a very important topic in NLP and many papers have been written that proposes new embedding techniques and compare different embedding techniques. Many papers have been written about improving a model's performance by swapping out parts of the model. In (Zhang and Ren, 2020), they proposed to use ELMo, which is an improved model over glove and word2vec techniques. Similarly to our approach to improve the performance of the LSTM by swapping out the embedding layer with a BERT embedding, they swapped the embedding layer of the BiDAF model out with ELMo embedding. Their reasoning for doing so is to "make the word representaion more consistent with the context semantics." As shown in their results, swapping the embedding out with ELMo increased the F1 score from 80.0 of the Gated Self-BIDAF to the 83.1 of the same model but with ELMo embedding. Similarly, we improved the performance of a LSTM model on various tasks and data sets by swapping out the one-hot-encoding layer of an vanilla LSTM to a BERT Embedding. We also found improved in performance when we replaced the embedding with Word2Vec, and GloVe. There are also attempts to replace the embedding int a BiDAF model with the state of the art BERT model. In (Sekhar and Mui), the authors mentioned that GloVe is a count based model and Word2Vec is a predictive model. The disadvantage of these models is that "a single word embedding is learned for each word and it cannot be adapted to the multiple meanings that a word might have."(Quoted from Related work section of (Sekhar and Mui)). They argue that BERT adds a task-specific top layer to a given model which should ideally help with the learning. Indeed their F1 score improved from 58 to 84.820 by swapping the embedding out to a BERT model.

Word2Vec(Goldberg and Levy, 2014) and GloVe embeddings(Pennington et al., 2014) are some of the most used embedding techniques in NLP models. The techniques used for the embedding varies. From this website, GloVe encodes co-occurrence probability ratio between two words as vector differences while Word2Vec is a predictive model that captures word embeddings as explained here. From this page, BERT uses context and handles polysemy and nuance better. It can be fine-tuned easily and is better at recognizing unknown words. It combines token embeddings, segment embeddings, and position embeddings and the input embedding is the sum of all the mentioned embedddings. It is a state-of-the-art model and we attempt to compare it to GloVe and Word2Vec embeddings to see if we can get a better performance similar to the findings in the other related works mentioned above.

As shown in figure 4, a word can have multiple meanings. Bank can refer to the financial bank or a river bank. This is a problem that occur in word2vec as explained here. Word2Vec embeds an word to one vector representation. This is a problem when a word could have different meanings in different contexts. BERT tries to solve this by considering the context. This may contribute to the improvements we see in the related works when the embedding layer is swapped out with BERT embedding.

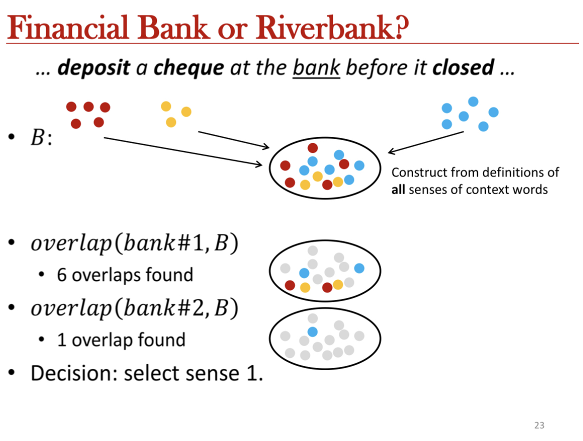
Financial Bank or Riverbank?
... deposit a cheque at the bank before it closed ...
• B:
Construct from definitions of all senses of context words
• $overlap(bank\#1, B)$
• 6 overlaps found
• $overlap(bank\#2, B)$
• 1 overlap found
• Decision: select sense 1.
23

Figure 4: From Comp 550 Lecture Slides. Borrowed from Jackie Cheung

3 Method

To test the effect of embedding on LSTM, we performed two type of experiments. To test out the differences of the embedding layers and to test whether a LSTM with BERT embedding performed the best out of all the embedding, we swapped out the embedding layer in the LSTM with Word2Vec, BERT, and GloVe embedding respectively. We then trained and ran each model on the same datasets. To try to test the effect of data on different embedding layers, we ran each model on several data sets. This is to test whether a embedding would perform better on one dataset than another. We hope to highlight the difference that the design choices of the embedding have on different types of data. We hope to answer the following questions; Does the size of dataset affect one type of embedding more than another? Which embedding do better on a data set with lesser used words?

3.1 Data Processing

For the covid dataset we tried to remove the @ symbol and emoji symbol because such symbol doesn't help the model learn from the dataset. We noticed that having emoji and other specialized symbols would our performance because it results in more unknown tokens which in return decreases performance. If the special symbols are connected to a string, the string would be invalidated. For example, a string like "@happy" would be invalidated because the whole string would be treated as a unknown token.

3.2 Models

For the experiments, we construted four models. We constructed our models in pytorch. Our vanilla LSTM model containing 4 layers (embedding, lstm, linear, and a dropout layer at the end), was used as the benchmark to measure against other models' performances. nn.Embedding is a built in embedding layer that is an lookup table from indices to vector. It is initialized to random weights and then trained during the training process. The model takes an input, applies embedding and then passes the embedded vector to an LSTM. We take the output of the embedding layer and pass it to torch.nn.Linear function which applies an linear transformation that allows us to do multi class or single class classification tasks.

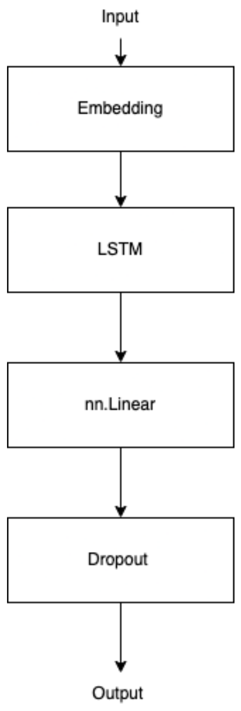

Figure 5: Model Structure

For the Word2Vec LSTM we removed the embedding layer and replaced it with an Word2Vec embedding. We loaded the pre-trained Word2Vec vector model from this website. We load the embedding by using the NN.Embedding.from_pre_trained function and inputting the loaded weights. We then make a embedding matrix and update the weight of the NN.Embedding layer with the weights of the new embedding matrix.

This procedure is similar for our other models.

Table 1: Clothing Dataset

	Word Embedding	Accuracy	Precision	Recall	F1
1	Vanilla	0.720	0.701	0.716	0.705
2	Word2Vec	0.647	0.615	0.647	0.628
3	Bert	0.634	0.606	0.634	0.616
4	Glove(50)	0.663	0.625	0.663	0.634
5	Glove(300)	0.659	0.618	0.659	0.623

Table 2: Spam Dataset

	Word Embedding	Accuracy	Precision	Recall	F1
1	Vanilla	0.971	0.971	0.971	0.971
2	Word2Vec	0.978	0.979	0.978	0.978
3	Bert	0.978	0.974	0.928	0.949
4	Glove(50)	0.950	0.949	0.950	0.949
5	Glove(300)	0.979	0.979	0.979	0.979

4 Datasets

We chosen several data sets for our experiments. The spam detentions data set contains 5169 unique messages. We chose this model because of it's simplicity. It is one of the smaller data sets with only 5000 data points. We want to see how our embedding perform with small data sets and small amounts of labels.

The Women's E-Commerce Clothing Reviews contains 23486 reviews and 5 labels with ratings. This dataset is more complicated and has more label categories. Using this dataset we wanted to see if the number of labels affects makes any difference on which embedding works best.

The covid-19-nlp-text-classification data set is a very small data set with 3798 tweets related to covid. We wanted to use this data set to test whether specialized words would cause certain embedding to perform worse than others. For example, the words 'sanitizer', 'coronavirus', 'masks' are words that grew in popularity only in the last 2 years. The pre-trained embedding we used are from many years ago so we will use this data set to see how if they perform well on data that they are not trained on.

We also calculated the Accuracy, Precision, Recall, and F1 of each model. We found that the descripancy between Word2Vec embedding which produces a 300 dimension embedding and the Glove embedding which produces a 50 dimension embedding was very high. Therefore we changed the GloVe embedding to produce a 300 dimension embedding to more fairly compare the GloVe and the Word2Vec.

Table 3: Covid Dataset

	Word Embedding	Accuracy	Precision	Recall	F1
1	Vanilla	0.640	0.640	0.641	0.639
2	Word2Vec	0.780	0.781	0.780	0.780
3	Bert	0.766	0.765	0.765	0.766
4	Glove(50)	0.661	0.622	0.661	0.628
5	Glove(300)	0.710	0.710	0.710	0.710

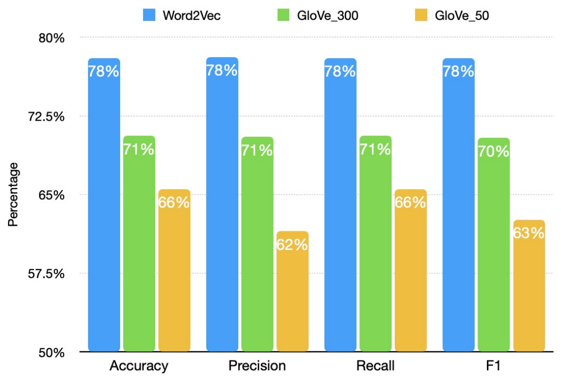

Figure 6: Comparison between evaluation accuracy of Word2Vec, GloVe_300, and Glove_50 on the COVID data set

5 Discussion and conclusion

5.1 Discussions About Results

From tables 1, 2, and 3, we see that swapping out the one hot encoding layer with another embedding layer improves the accuracy of the LSTM model. We see however, that adding GloVe Embedding to the LSTM on the spam detection actually decreased the accuracy from an Vanilla LSTM. Furthermore, we noticed that Glove Embedding performs worse than Word2Vec embedding on all the datasets we tested. This result closely resembles the results of (Baroni et al., 2014). We validated our hypothesis by showing that the BERT embedding performed better than all other embedding and encoding techniques on every data set. This may be due to the Masked Language Modelling and NSP techniques. By masking a part of a sentence using the MLM technique, it forces the BERT to learn to consider the context of a sentence as explained here. Further context is learned by the BERT using next sentence prediction techniques. This forces the BERT to learn the context and relationship between sentences. NSP is explained in more detail here. Thus having both the awareness of the relationship between sentences and words within a sentence, BERT has the ability to overcome the issue show previously in figure 4. Word2Vec is also able to learn the contextual information and hence it also has a high accuracy like the BERT on most data sets we tried. Unlike Word2Vec, glove using the global word counts to find calculate the embedding, hence it is not able to learn as much local context as GloVe which is representative of it's lower accuracy when compared to Word2Vec and BERT. However, we observed that glove embedding performed worse than the vanilla lstm on the spam detection dataset. This need further investigation. Another observation is that the different in accuracy between the Word2Vec LSTM and Glove increases as we decrease the size of the datasets. We also see that the performance of all lstm models decreases as we run them on smaller datasets. This is consistent with other models. If a model has less info to learn from, then it is logical that the model would have a lower accuracy since it wasn't able to learn from as much data.

6 Limitations

We were able to demonstrate and verify our hypothesis that BERT, with it's ability to learn from contextual information, would not only outperform other models but a BERT embedding would also outperform other embedding techniques as well. However, due to the restraint of time and resources we did not get the chance to try other more complicated model structures. According to (Sekhar and Mui) training a BiDaf took days if not hours on state of the art machines. It would take a lot of time and resources to train it. In (Sekhar and Mui) they only tested Bidaf with BERT embeddding. However, we feel like it would be interesting to see if it is possible to use other embedding techniuqes as well. Furthermore, we can try the experiement done in (Sekhar and Mui) but using our methodology of running it on different data sets with different qualities to see how embedding gets affected by differences in data. We also implemented a LSTM with BERT Sentence Embedding instead of just a BERT Word embedding since that would allow us to take advantage of BERT's ability to learn contextual relationships. However, we were restrained by the amount of ram availiable to us.

7 Statement of contributions

- Javin Liu - Latex Report, BERT Word Embedding LSTM on Spam Detection, BERT Word Embedding on Clothing Review
- Yuxuan Tian - Vanilla LSTM on Clothing Review , Word2Vec LSTM
- Zhixin Xiong - Glove LSTM on Covid Dataset, Glove LSTM on Spam Detection

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *International conference on artificial neural networks*, pages 799–804. Springer.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Priyanka Sekhar and Ryan Mui. Enhancing BiDAF with BERT Embeddings, and Exploring Real-World Data.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Weiwei Zhang and Fuji Ren. 2020. Elmo+ gated self-attention network based on bidaf for machine reading comprehension. In *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, pages 1–6. IEEE.